CLAIMS: *The following is a listing of all claims in the application with their status and the text for all active claims.*

1. (CANCELED)
2. (CANCELED)
3. (CANCELED)
4. (CANCELED)
5. (CANCELED)
6. (CANCELED)
7. (CANCELED)
8. (CANCELED)
9. (CANCELED)
10. (CANCELED)
11. (CANCELED)
12. (CANCELED)
13. (CANCELED)
14. (CANCELED)
15. (CANCELED)
16. (CANCELED)
17. (CANCELED)
18. (CANCELED)
19. (CANCELED)
20. (CANCELED)
21. (CANCELED)
22. (CANCELED)
23. (CANCELED)
24. (CANCELED)
25. (CANCELED)
26. (CANCELED)
27. (CANCELED)
28. (CANCELED)
29. (CANCELED)
30. (CANCELED)
31. (CANCELED)
32. (CANCELED)
33. (NEW) A computer-implemented method for improving compression for storage of a plurality of parallel data element sequences comprising:

(a) creating a dictionary of unique values for each of said data element sequences, wherein each dictionary associates a numeric index with each unique value,

(b) forming an n-ary tree with leaf and interior nodes wherein:

(1) each said leaf node corresponds to one of said dictionaries,

(2) each said interior node associates a numeric index with tuples of numeric indexes from other subordinate leaf or interior nodes,

(3) one or more interior nodes are capable of storing one or more sequences of mutually-consecutive tuples by representing said sequences in a form that uses less storage space than representing said sequences as individual tuples, and

(4) one or more interior nodes are capable of:

i. recording the addition of a tuple that extends a tuple sequence by modifying one or more fields in the representation of said sequence that are capable of representing the length of said sequence, or

ii. recording the addition of a tuple that invalidates an existing tuple sequence by splitting said tuple sequence into one or more subsequences, wherein none of the tuples of said subsequences contain any element of said added tuple, or

iii. recording the addition of a tuple that has not been previously added to said interior node, wherein said added tuple does not extend a tuple sequence, by adding said tuple to a tuple collection, or

iv. any combination of two or more of i, ii, and iii,

wherein the forming step comprises:

(c) defining a problem space comprising:

(1) a set of states such that each said state contains a set of leaves and zero or more interior nodes, each with zero or more other nodes as children, and

(2) a value function, giving a numeric ranking of the value of any state's design,

(d) defining one or more operators that transform one state to another, and

(e) searching the problem space, starting from an initial state and applying operators to move to other states until a state with an acceptable n-ary tree design is reached.

34. (NEW) The method of claim 33, wherein said method for arranging an n-ary tree uses an estimate of interior node size, from a function of the sizes of said interior node's child nodes.

35. (NEW) The method of claim 33, wherein said tree-arrangement method uses an operator that joins two or more nodes (leaf nodes or interior nodes) under an interior node.

36. (NEW) The method of claim 33, wherein each unique value of a leaf node or each unique tuple of an interior node is capable of being associated with a count of the number of times that value or tuple of values occurred in the parallel data element sequences.

37. (NEW) The method of claim 33, wherein said method is used to compress a table, wherein said parallel data element sequences represent fields of said table's records.

38. (NEW) The method of claim 33, wherein said value function gives a value which indicates whether a state's n-ary tree design is acceptable.

39. (NEW) A computer-implemented method for improving compression for storage of a plurality of parallel data element sequences comprising:

(a) creating a dictionary of unique values for each of said data element sequences, wherein each dictionary associates a numeric index with each unique value,

(b) forming one or more n-ary trees with leaf and interior nodes wherein:

(1) at least one of said leaf nodes is distinct from, and represents a subset of values from one of said dictionaries,

(2) each interior node associates a numeric index with tuples of numeric indexes from other subordinate leaf or interior nodes,

wherein the forming step comprises:

(c) defining a problem space comprising:

(1) a set of states such that each said state contains a set of leaves and zero or more interior nodes, each with zero or more other nodes as children, and

(2) a value function, giving a numeric ranking of the value of any state's design,

(d) defining one or more operators that transform one state to another, and

(e) searching the problem space, starting from an initial state and applying operators to move to other states until a state with an acceptable n-ary tree design is reached.

40. (NEW) The method of claim 39, wherein said method for arranging an n-ary tree uses an estimate of interior node size, from a function of the sizes of said interior node's child nodes.

41. (NEW) The method of claim 39, wherein said tree-arrangement method uses an operator that joins two or more nodes (leaf nodes or interior nodes) under an interior node.

42. (NEW) The method of claim 39, wherein each unique value of a leaf node or each unique tuple of an interior node is capable of being associated with a count of the number of times that value or tuple of values occurred in the parallel data element sequences.

43. (NEW) A computer-implemented method for storage of a plurality of parallel data element sequences, and efficiently processing elements from a subset of said sequences, comprising:

(a) creating a dictionary of unique values for each of said data element sequences, wherein each dictionary associates a numeric index with each unique value,

(b) forming one or more n-ary trees with leaf and interior nodes wherein:

(1) each leaf node corresponds to one of said dictionaries,

(2) each interior node associates a numeric index with tuples of numeric indexes from other subordinate leaf or interior nodes,

(3) a gate field is defined for one or more interior nodes,

(c) processing the leaves corresponding to said subset of sequences by:

(1) setting the value of said gate field for each said interior node, to indicate which of said interior node's branches lead to leaf nodes in said subset,

(2) following paths that lead to said leaf nodes, and

(3) processing said elements in said leaf nodes encountered,

wherein the forming step comprises:

(d) defining a problem space comprising:

(1) a set of states such that each said state contains a set of leaves and zero or more interior nodes, each with zero or more other nodes as children, and

(2) a value function, giving a numeric ranking of the value of any state's design,

(e) defining one or more operators that transform one state to another, and

(f) searching the problem space, starting from an initial state and applying operators to move to other states until a state with an acceptable n-ary tree design is reached.

44. (NEW) The method of claim 43, wherein said method for arranging an n-ary tree uses an estimate of interior node size, from a function of the sizes of said interior node's child nodes.

45. (NEW) The method of claim 43, wherein said tree-arrangement method uses an operator that joins two or more nodes (leaf nodes or interior nodes) under an interior node.

46. (NEW) The method of claim 43, wherein each unique value of a leaf node or each unique tuple of an interior node is capable of being associated with a count of the number of times that value or tuple of values occurred in the parallel data element sequences.

47. (NEW) The method of claim 43, where said processing includes using values or tokens at said leaf nodes to reconstruct a subset of a stored record.

48. (NEW) The method of claim 47, further including the step of adding one or more of said reconstructed record subsets to another tree.

49. (NEW) The method of claim 33 wherein at least one of said leaf nodes is distinct from, and represents a subset of values from one of said dictionaries.

50. (NEW) The method of claim 33, further including a method for efficiently processing elements from a subset of said parallel data element sequences, comprising:

(a) defining a gate field for one or more interior nodes,

(b) processing the leaves corresponding to said subset of sequences by:

(1) setting the value of said gate field for each said interior node, to indicate which of said interior node's branches lead to leaf nodes in said subset,

(2) following paths that lead to said leaf nodes, and

(3) processing said elements in said leaf nodes encountered.

51. (NEW) The method of claim 39, further including a method for efficiently processing elements from a subset of said parallel data element sequences, comprising:

(a) defining a gate field for one or more interior nodes,

(b) processing the leaves corresponding to said subset of sequences by:

(1) setting the value of said gate field for each said interior node, to indicate which of said interior node's branches lead to leaf nodes in said subset,

(2) following paths that lead to said leaf nodes, and

(3) processing said elements in said leaf nodes encountered.

52. (NEW) The method of claim 49, further including a method for efficiently processing elements from a subset of said parallel data element sequences, comprising:

(a) defining a gate field for one or more interior nodes,

(b) processing the leaves corresponding to said subset of sequences by:

(1) setting the value of said gate field for each said interior node, to indicate which of said interior node's branches lead to leaf nodes in said subset,

(2) following paths that lead to said leaf nodes, and

(3) processing said elements in said leaf nodes encountered.